Improving Demonstration Diversity by Human-Free Fusing for Text-to-SQL

Anonymous ACL submission

Abstract

Currently, the in-context learning method based on large language models (LLMs) has become the mainstream of text-to-SQL research. Previous works have discussed how to select demonstrations related to the user question from a human-labeled demonstration pool. However, human labeling suffers from the limitations of insufficient diversity and high labeling overhead. Therefore, in this paper, we discuss how to measure and improve the diversity of the demonstrations for text-to-SQL. We present a metric to measure the diversity of the demonstrations and analyze the insufficient of the existing labeled data by experiments. Based on the above discovery, we propose fusing iteratively for demonstrations (FUSED) to build a high-diversity demonstration pool through human-free multiple-iteration synthesis, improving diversity and lowering label cost. Our method achieves an average improvement of 3.2% and 5.0% with and without human labeling on several mainstream datasets, which proves the effectiveness of FUSED.[1]

1 Introduction

Text-to-SQL is an important task that garners widespread attention for reducing the overhead of accessing databases by automatically generating SQL queries in response to user questions (Qin et al., 2022). Currently, in-context learning based on large language models (LLMs) has become the predominant approach to the text-to-SQL task, which can significantly enhance performance while reducing the need for fine-tuning (Pourreza and Rafiei, 2023; Nan et al., 2023; Chang and Fosler-Lussier, 2023a). Regarding in-context learning for text-to-SQL, in addition to the user question and the database, the LLM is also provided with several demonstrations, guiding the LLM in accurately generating SQL corresponding to the user question.

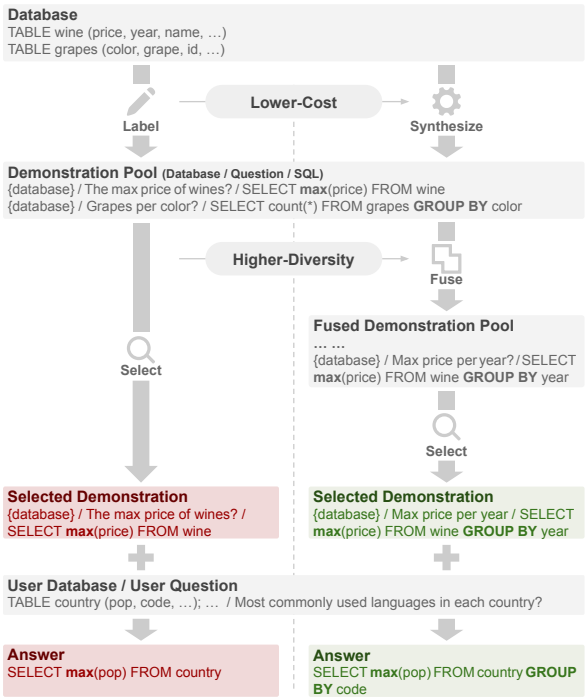

Figure 1: The comparison between the baseline (left) and FUSED (right) of obtaining the demonstration pool for text-to-SQL.

Currently, there are many works (Chang and Fosler-Lussier, 2023b; Su et al., 2023; Luo et al., 2024) explore how to select demonstrations relevant to user questions from a human-labeled demonstration pool. However, in such works, the demonstration pool that relies entirely on human labels limits the performance of text-to-SQL based on in-context learning because of two problems: *1.* Regarding quality, human-labeled data has shortcomings in the diversity (Ramalingam et al., 2021; Guo, 2023). *2.* Considering the cost, human labeling demands a high labor overhead. To solve the above problems, enhancing text-to-SQL performance, in this paper, we discuss: ***1. Theoretically, how to measure the diversity of the demonstration pool*** (§2); ***2. Practically, how to build a diverse demonstration pool without human labeling*** (§3).

[1] Our data and code will be released after review.

First, we discuss the diversity insufficient of human labeling from the perspective of theoretical analysis. We first discuss the necessity of the demonstration of diversity, of which we present a formal definition. Then, based on this definition, we put forward to measure the diversity to prove the diversity insufficient of existing text-to-SQL labeled data, demanding enhancing data diversity.

Based on the analysis above, we present **FUS**ing it**E**ratively for **D**emonstrations (FUSED), which synthesizes demonstrations iteratively using LLMs. An illustration of our method is shown in Figure 1. About the problem of high labeling cost, our method employs LLMs to synthesize demonstrations, reducing the human labeling overhead. For the problem of low diversity, in each iteration, FUSED fuses the demonstrations of the previous iteration, ensuring that the fused demonstrations are dissimilar from the previous demonstrations, thereby improving the diversity.

To prove the effectiveness of our method, we adapt FUSED to several mainstream text-to-SQL datasets: Spider (Yu et al., 2018) and KaggleDBQA (Lee et al., 2021). The experimental results show that our method brings an average improvement of 3.2% and 5.0% with and without labeling data, proving the effectiveness of our method. Further analysis experiments show that FUSED effectively improves the metric we present, demonstrating that our method can indeed enhance the diversity of the demonstration pool.

The contributions of our work are as follows:

- To theoretically analyze the diversity of the existing labeled demonstrations for text-to-SQL, we present a metric to measure the diversity, proving that the insufficiency of the diversity.
- To practically obtain a high-diversity demonstration pool without human labeling, we propose FUSED, which enhances the diversity by human-free synthesis with multiple iterations.
- To validate FUSED, we adapt our method on multiple mainstream text-to-SQL datasets, which achieves 3.2% and 5.0% performance improvements with and without human labeling, demonstrating the effectiveness of our method.

2 Analysis: Insufficient Diversity of Labeled Text-to-SQL Demonstrations

In this section, we present that **the diversity of the existing labeled text-to-SQL demonstrations is insufficient**. First, we discuss the necessity of the high diversity of the demonstration pool, of which we present a formal definition. Then, we present a metric for measuring the diversity of the demonstration pool and explore the insufficient labeling diversity with the experiment results.

2.1 Necessity of Diversity

About the in-context learning, since LLMs imitate the provided demonstrations to generate the answer (Xun et al., 2017), it is required to ensure the selected demonstration is as similar to the user question as possible. However, the user question is unpredictable, leading to that the demonstration pool should contain as diverse demonstrations as possible to cover various user questions. That is, for any user question, there should be a demonstration with no difference from the user question.

Formally, we define u as the user question and $D = \{d_i\}$ as the demonstration pool, where d_i is each demonstration. The above analysis can be summarized as that D should satisfy Equation 2.1, where `diff` denotes the difference between the demonstration and the user question (e.g., SQL structure, domain knowledge). We discuss how to calculate `diff` in Appendix A.

$$\max_{u} \min_{d_i \in D} \texttt{diff}(d_i, u) = 0 \tag{2.1}$$

2.2 Insufficient of Labeling Diversity

To discuss whether the diversity of the existing text-to-SQL labeling demonstrations is sufficient, in this part, we present a metric to measure the diversity. Although the demonstration is hard to be the same as the user question since it is unpredictable, to ensure the performance, the difference between the user question and the demonstration should be as small as possible, which satisfaction can be formally represented as Equation 2.2.

$$D^* = \operatorname*{argmin}_{D} \max_{u} \min_{d_i \in D} \texttt{diff}(u, d_i) \tag{2.2}$$

From the equation, it can be seen that the expression $\max_u \min_{d_i \in D} \texttt{diff}(u, d_i)$ in the `argmin` changes only depending on D and determines whether D satisfy Equation 2.1. Therefore, we use Equation 2.3 to measure the diversity of the demonstration pool, which we call *diversity measurement*, where taking the multiplicative inversion is to make this metric increase as the diversity increases. We discuss how to calculate the diversity measurement in Appendix A.

$$\text{DM} = (\max_{u} \min_{d_i \in D} \texttt{diff}(u, d_i))^{-1} \tag{2.3}$$

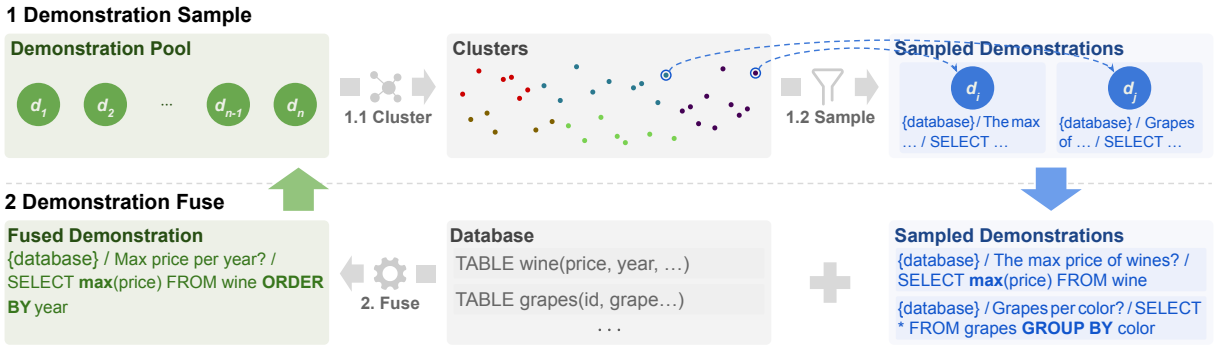

Figure 2: The pipeline of FUSED, which consists of two steps: **1. Demonstration Sample**: Sample demonstrations to be fused from the demonstration pool; **2. Demonstration Fuse**: Fuse the sampled demonstrations. The representation of `{database}` is discussed in Appendix C.

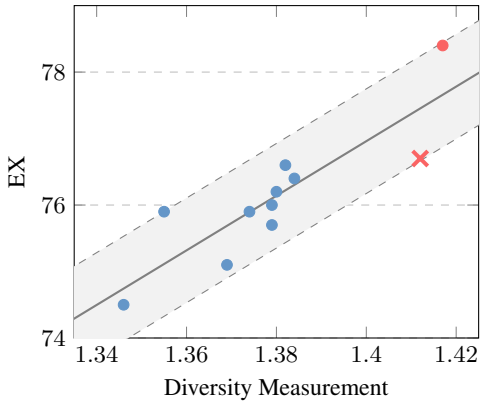

Figure 3: The performance with the change of diversity measurement on Spider. x denotes the original label data of Spider and • denotes the data under different synthesized scales introduced in §3. Blue denotes human-free synthesized data, and red denotes that is based on human-labeled data.

With the metric to measure the diversity, we then discuss the diversity of the existing text-to-SQL labeling demonstrations. The performance of the demonstration pool with different diversity measurements is shown in Figure 3. From the figure, we can see that, although the diversity of labeled data is relatively high, the diversity can still be improved and insufficient. Therefore, in the following, we discuss how to synthesize demonstrations to improve the diversity of the demonstration pool.

3 Method

Our method focuses on how to synthesize new demonstrations given databases to improve the diversity. Considering the poor diversity of directly generating demonstrations only relying on the sampling generation (Cegin et al., 2024), we present to improve the diversity of the demonstration pool by fusing different demonstrations iteratively. In each iteration, we guide the model to generate demonstrations that are not similar to the previous iterations, thereby improving the diversity. An illustration of FUSED is shown in Figure 2.

A simple example of our method is clustering the demonstrations based on the SQL keywords (e.g., `WHERE`, `ORDER BY`). Then, we sample and fuse demonstrations from the clusters corresponding to keywords `WHERE` and `ORDER BY`. The fused demonstration contains both `WHERE` and `ORDER BY` that are different from the sampled demonstrations, improving the demonstration diversity.

3.1 Overview

The fusion process of FUSED starts with an initial demonstration pool, which can be human-labeled or synthesized by LLMs (see Appendix B). FUSED includes multiple iterations of fusion, where the synthesis of each iteration is based on the demonstration pool of the previous iteration. Each iteration consists of *demonstration sample* (§3.2) and *demonstration fuse* (§3.3) two steps, which sample and fuse the demonstrations of the demonstration pool separately. The fused demonstrations of each iteration are then added to the demonstration pool, preparing for the next iteration.

After all iterations of fusion, we use the final demonstration pool for the text-to-SQL based on the in-context learning. We generate the SQL of each user question with LLMs directly following Chang and Fosler-Lussier (2023b) since this is not the main topic of this paper.

3.2 Demonstration Sample

This step is to sample the demonstrations to be fused, which consists of: *1.* Cluster: dividing the demonstrations into multiple clusters; *2.* Sample: sampling demonstrations from clusters to be fused.

3.2.1 Cluster

Before the fusion to get new demonstrations, it is required that the demonstrations sampled for fusing are not similar to ensure that the fused demonstration is not similar to the sampled demonstrations, thereby enhancing the diversity. The previous work (Zhang et al., 2023b) has shown that similar demonstrations are in the same cluster after encoding and then clustering. That is because the encoded vectors can reflect the attributes of the demonstrations (e.g., SQL structure, domain knowledge), where the closer the vector distance, the more similar the attributes.

Inspired by this, we empirically use Sentence-BERT (Reimers and Gurevych, 2019) to encode the concatenation of the question and the SQL of all demonstrations in the pool, and then use K-means to cluster encoded results into multiple clusters. Compared with not using the cluster, our method can ensure that the corresponding encoding vectors of the sampled demonstrations from different clusters are far away, leading to the demonstration used for fusion is not similar and diverse.

3.2.2 Sample

After obtaining different clusters of the demonstration pool, we then sample demonstrations from different clusters for fusing. Considering that even in the same cluster, the demonstrations are not the complete same, such as domain knowledge and question semantics, which could affect the similarity with the user question. To improve diversity, during the demonstration sampling, we randomly choose several clusters, and then randomly sample demonstrations from each cluster separately, making the fused demonstration reflect the difference between different demonstrations.

3.3 Demonstration Fuse

We employ LLM synthesize to fuse demonstrations as the discussion in Appendix B, where we add the sampled demonstrations to guide the synthesis as in-context learning. Adding the sampled demonstrations comes up because LLMs imitate the demonstrations to generate results, whereas the fused demonstration is generated by imitating the sampled demonstration, thereby achieving the fusion effect. Thus, the fused demonstrations can reflect the attributes of and be different from all sampled demonstrations, thereby improving the diversity of the demonstration pool.

4 Experiments

4.1 Experiment Setup

Dataset We evaluate FUSED on two text-to-SQL datasets: Spider (Yu et al., 2018) and KaggleDBQA (Lee et al., 2021). Spider, a multi-domain text-to-SQL dataset, is one of the most widely used datasets currently. KaggleDBQA is smaller in scale but involves more complex database and SQL structures, presenting higher hardness. In the following, for simplicity, we refer to KaggleDBQA as Kaggle.

Metric Following previous works (Yu et al., 2018; Pourreza and Rafiei, 2023; Li et al., 2023), we employ execution match (EX) as our evaluation metric. EX measures the accuracy by comparing the execution results of the generated SQL on the database. There are two ways to evaluate EX: directly using the predicted SQL conditional value (with value) and using the conditional value in the correct answer (without value).

Model In our experiments, we use CodeLlama (Rozière et al., 2023) and GPT3.5 [2] to synthesize demonstrations and convert user questions into SQLs. We also apply FUSED to ACT-SQL (Zhang et al., 2023a) and ODIS (Chang and Fosler-Lussier, 2023b). The detailed introduction of the above models can be seen in Appendix D.

Implementation Details About KaggleDBQA, since it only contains 8 databases, we use Spider databases to synthesize demonstrations. For each database, we generate 8 SQLs separately, set the generation temperature to 0.3, and synthesize in turns of 3. The prompts for synthesizing demonstrations and text-to-SQL are shown in Appendix C.

4.2 Main Result

The main experimental results are shown in Table 1, where FUSED brings 2.6% performance improvement on average across different settings, showing the effectiveness of our method. Besides, from the table, we can also see that:

Model Scale Our method brings significant performance improvements on models of different scales. However, our method brings performance degradation with CodeLlama-7b, because of the low quality of the synthesized demonstrations due to its relatively poor performance.

[2] https://platform.openai.com/docs/models/gpt-3-5

Dataset	Prompt	CodeLlama 7b w.	CodeLlama 7b w/o.	CodeLlama 13b w.	CodeLlama 13b w/o.	CodeLlama 34b w.	CodeLlama 34b w/o.	GPT3.5 - w.	GPT3.5 - w/o.	Δ - w.	Δ - w/o.
Spider	Zero	48.5	59.8	54.9	67.6	56.9	72.2	57.9	74.9	+3.4	+3.4
	+ FUSED	54.4	66.4	58.8	70.9	59.7	75.1	58.7	75.8		
	Label	55.3	67.5	58.8	72.1	61.6	76.7	61.6	80.3	+1.6	+1.4
	+ FUSED	56.8	69.0	60.4	74.2	63.2	78.4	63.2	80.7		
	ACT-SQL†	62.1	63.2	67.5	69.1	71.0	72.8	75.8	77.6	+0.7	+0.9
	+ FUSED	60.3	61.7	68.4	69.8	74.6	76.7	76.0	78.0		
	ODIS†	58.2	71.8	61.9	76.6	64.3	80.9	63.9	81.1	+0.8	+0.9
	+ FUSED	58.0	71.0	62.9	78.0	65.6	82.1	64.8	83.0		
Kaggle	Zero	9.9	18.0	13.2	23.5	13.2	23.2	14.0	25.4	+6.1	+7.2
	+ FUSED	22.8	32.0	19.1	29.0	18.0	30.1	14.7	27.6		
	Label	27.9	39.7	32.4	44.1	26.5	38.6	26.5	40.4	+5.4	+4.2
	+ FUSED	35.3	47.1	34.6	46.0	32.4	45.6	32.4	40.8		
	ACT-SQL†	27.6	30.5	30.5	33.8	33.8	38.2	29.4	31.6	+0.4	+0.5
	+ FUSED	27.6	30.9	30.5	33.8	33.8	38.6	30.9	32.7		
	ODIS†	33.8	43.4	34.6	47.1	31.6	46.3	34.6	48.9	+2.3	+3.0
	+ FUSED	35.7	47.1	36.0	48.5	35.3	50.4	36.8	51.5		

Table 1: The main experimental results of FUSED. Zero denotes zero-shot inference, and Label denotes using human-labeled data. About the metric, w. denotes with values and w/o. denotes without values. † denotes the reproduction results by us since the performance differences brought by the API version of GPT3.5. The improved results led by FUSED are marked green, performance degradation is marked in red, and unchanged results are marked in black. The best results of different models and datasets are annotated in underline. **Δ** denotes the average improvement of different prompt methods leading by FUSED.

Model Type Our method continues to improve performance based on the labeled data and two well-designed methods (ACT-SQL and ODIS) under most settings, proving the generalization and effectiveness of FUSED. In addition, the results also prove that with a fixed demonstration selection method, modifying the demonstration pool can further enhance the performance.

Dataset FUSED brings significant performance improvements on all experimental datasets and even achieves results close to labeled data on Spider under the zero setting, demonstrating the effectiveness of our method under different domains. Besides, our method achieves more significant improvement on KaggleDBQA than Spider, showing that the demonstrations synthesized by FUSED are more effective for complex text-to-SQL questions.

4.3 Ablation Studies

To verify the effectiveness of the iteration and the cluster designed by our method, we perform ablation experiments on each part separately. The experimental results are shown in Table 2. Based on such results, we discuss the impact of different parts on the performance of our method.

Dataset	Prompt	7b	13b	34b
Spider	FUSED	66.4	70.9	75.1
	- Iteration	66.2(−0.2)	69.9(−1.0)	73.9(−1.2)
	- Cluster	65.3(−1.1)	69.9(−1.0)	74.6(−0.5)
Kaggle	FUSED	32.0	29.0	30.1
	- Iteration	30.1(−1.9)	28.8(−0.2)	28.7(−1.4)
	- Cluster	26.5(−5.5)	26.5(−2.5)	30.0(−0.1)

Table 2: EX without values of FUSED under the ablation of validation, cluster, and iteration with CodeLlama without human-labeled data.

4.3.1 Ablation of Iteration

To demonstrate that iterations work by improving the quality rather than quantity of the demonstrations, we conduct experiments that generate the same number of data as our method without iterations. From Table 2, we can see that: *1.* FUSED achieves significant performance gains compared with generation without iterations, proving that our method indeed enhances the model performance by improving the quality of the generated demonstrations. *2.* For larger-scale models, iteration has a more significant impact on performance, indicating that larger-scale models can more effectively synthesize diverse demonstrations through multiple iterations, improving performance more effectively.

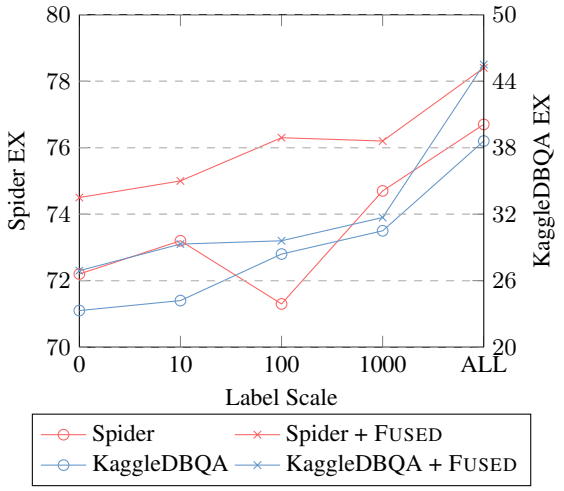

Figure 4: The EX without values of CodeLlama-34b with and without FUSED under different initial labeling scales. The X-axis represents the labeled data scale used for synthesis. The Y-axis on the left and right represent the results of Spider and KaggleDBQA respectively.

4.3.2 Ablation of Cluster

To demonstrate the effectiveness of the cluster, we perform ablation experiments on it. We compare our method with randomly selecting demonstrations during the demonstration fuse step. From Table 2, we can find: *1*. Synthesis without clustering brings performance degradation in all settings, proving the effectiveness of the cluster. *2*. The performance degradation of KaggleDBQA is more obvious compared to Spider, indicating that the more complex text-to-SQL questions are more sensitive to the demonstration diversity.

4.4 Analysis

In this part, we adapt analysis experiments to discuss how different factors affect the performance of FUSED. The reason for the experimental settings we used can be seen in Appendix E.

4.4.1 Turn Number

To analyze the effectiveness of the iteration, we adapt experiments with different iterative turns, which are summarized in Figure 5 and Figure 6. From the table, we can see that: *1*. When turn ≤ 4, as the turn increases, diversity measurement and the performance of our method improve steadily, indicating that multiple iterations can enhance the diversity, thereby enhancing performance. *2*. When turn > 4, with the number of turns increasing, diversity and performance improvement brought by FUSED becomes less and less, indicating the diversity can not be infinitely improved.

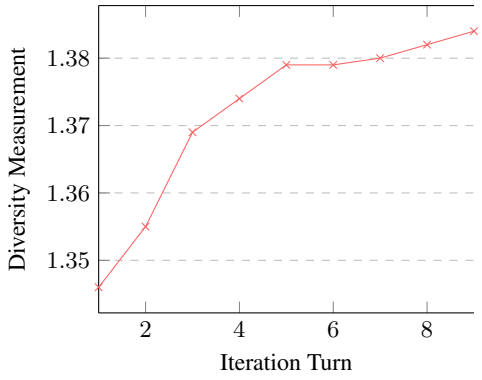

Figure 5: Diversity measurement of CodeLlama-34b across different iteration turns with FUSED.

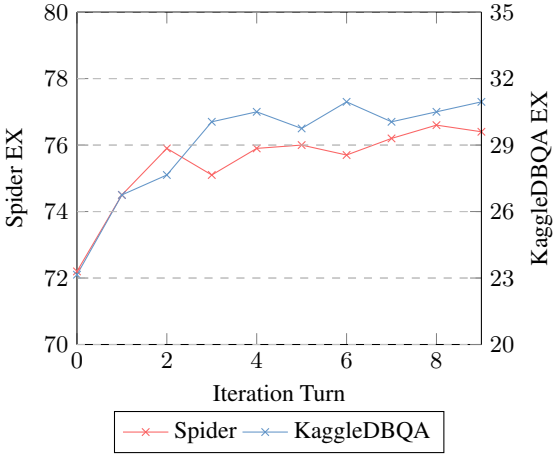

Figure 6: EX without values of CodeLlama-34b across various turns with FUSED. The X-axis denotes the turns of FUSED. The Y-axis on the left and right represent the results of Spider and KaggleDBQA respectively.

4.4.2 Label Scale

Although the main experiments of Table 1 demonstrate the effectiveness of our method on labeled data, the practical applications could lack labeled data with the same scale as the Spider training data. Therefore, to validate the effectiveness of FUSED across varying scales of labeled data, we randomly sample different scales of initial labeled data from Spider training data and conduct experiments on these subsets. The experimental results with different labeling scales are present in Figure 4.

From the figure, we can see that: *1*. Under most settings, our method brings performance improvement, indicating its widespread effectiveness under different initial label scales. *2*. With the increase of the initial label scale, the performance demonstrates a consistent upward trend, suggesting that expanding the scale of label scale can reliably enhance model capabilities.

Database	7b	13b	34b
None	18.0	23.5	23.2
Kaggle	29.0	24.3	27.6
Kaggle + Spider	32.0	29.0	30.1

Table 3: EX without values of FUSED using CodeLlama evaluated on KaggleDBQA with different synthesizing databases. **Database** denotes the databases used for synthesizing. None denotes not synthesizing, Kaggle denotes only using the KaggleDBQA databases, and Kaggle + Spider denotes mixing Spider databases.

Dataset	Prompt	Easy	Medium	Hard	Extra
Spider	Zero	**88.7**	80.7	57.5	40.4
	FUSED	87.5	**81.2**	**63.8**	**52.4**
Kaggle	Zero	53.1	30.3	5.1	1.9
	FUSED	**59.4**	**32.9**	**11.4**	**1.9**

Table 4: EX without values of CodeLlama-34b under different SQL hardness with and without FUSED. Zero denotes results under the zero-shot setting. The best result of each setting is annotated in **bold**.

4.4.3 Database Domain

In this part, we evaluate that FUSED can improve the text-to-SQL performance across different domains without human labeling. We only use the databases of KaggleDBQA to synthesize demonstrations for KaggleDBQA, while we use the Spider databases in the main experiment following the previous work (Chang and Fosler-Lussier, 2023b). Since KaggleDBQA only has 8 databases, for each database, we generate 128 SQLs to ensure to obtain high diverse demonstrations.

The experimental results are shown in Table 3. From the table, we can see that: *1*. Compared with not synthesizing demonstrations, FUSED can bring performance improvements when only using KaggleDBQA databases, proving the effectiveness of our method adapted to a new domain without labeling. *2*. Compared to using only KaggleDBQA databases, the demonstrations obtained by mixing Spider databases can bring greater performance improvements, indicating that increasing the diversity of databases can also improve the diversity of synthesized demonstrations.

4.4.4 SQL Hardness

To analyze the effectiveness of FUSED on questions with different complexity, we evaluate our method on SQL categorized by different hardness. The category criteria follows Yu et al. (2018). The experimental results are shown in Table 4.

Template (%)
`SELECT * FROM * WHERE * <op> *` (25.7)
`SELECT * FROM * WHERE * <op> * AND * <op> *` (13.9)
`SELECT * FROM * JOIN * JOIN * WHERE * <op> *` (5.2)
`SELECT * FROM * JOIN * WHERE * <op> *` (4.9)
`SELECT * FROM * WHERE * IN (SELECT * FROM * WHERE * <op> *)` (4.3)

Table 5: Top five SQL templates synthesized by FUSED using CodeLlama-34b. The numbers in the brackets denote the proportion of each template.

From the table, we can see that: *1*. On most hardness, our method can bring significant performance improvements, which proves the effectiveness of FUSED. *2*. On Spider, the more difficult SQL, the more significant the improvement, showing that synthesized demonstrations can more effectively guide complex SQL generation. *3*. For the easy questions of Spider, our method brings a slight performance degradation because the model already performs well under the zero-shot setting for this hardness, and the additional demonstrations could mislead the model. *4*. On the extra questions of KaggleDBQA, our method does not bring performance improvement, which could be because it is too hard to synthesize too complex demonstrations (harder than Spider extra questions), resulting in the selected demonstrations being unable to effectively guide the generation of the extra hardness.

4.4.5 Synthesized Template

To guide future works in generating more diverse demonstrations, in this part, we analyze the proportion of demonstrations with different SQL templates synthesized by our method. We replace table names, column names, and values with * and operators with <op> as the templates corresponding to each SQL. Our method synthesizes 175 different SQL templates, showing the diversity of the synthesized demonstrations. The five most frequent template types are shown in Table 5.

From the table, we can find: *1*. The current model is most inclined to generate `SELECT` and `WHERE`, which is nearly 40%, indicating that such types of SQL occur more frequently in the pretraining data of LLMs we use and, thereby, are more frequently used in real-world scenarios. *2*. Existing models hardly generate complex SQL that contains nested SQL (less than 5% of synthetic data), indicating that future methods should specifically pay attention to guide the model to generate results that contain two or more sub-SQLs or even more complex structures.

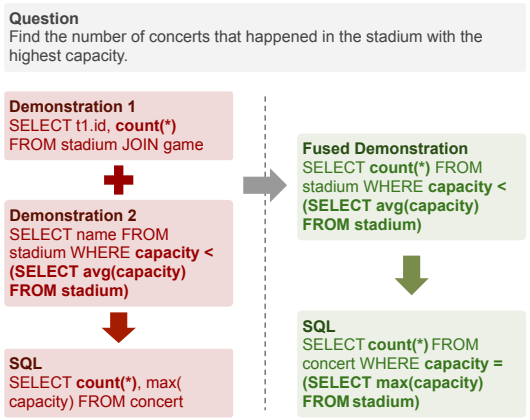

Figure 7: The case study of demonstrations by human-labeling (left) and FUSED (right) from Spider. The corresponding SQL keywords between demonstrations and the answer are annotated in **bold**.

4.4.6 Case Study

Although the above analysis proves the effectiveness of FUSED, how our method improves the performance of the text-to-SQL using in-context learning remains to be discovered. In order to analyze how our method improves the model performance more specifically, in this part, we conduct a case study. A comparison between results based on labeled data and the demonstrations obtained using FUSED is shown in Figure 7.

From the figure, we can see that the results using only labeled data do not combine the SQL keywords of the two demonstrations well. The demonstration obtained with our method, on the other hand, has already combined the SQL keywords of the two demonstrations, which guides the model to successfully generate the correct SQL.

5 Related Works

5.1 Text-to-SQL

Text-to-SQL is a vital task that generates SQL based on the user question and the provided databases. Recent research shows that text-to-SQL based on LLMs can approach or exceed the performance of fine-tuned models without fine-tuning, which greatly advances research on this task while reducing labeling overhead (Chang and Fosler-Lussier, 2023b; Zhang et al., 2023a; Li and Xie, 2024). For example, DIN-SQL (Pourreza and Rafiei, 2023) decomposes the text-to-SQL task into multiple sub-tasks and solves these sub-tasks separately. DAIL-SQL (Gao et al., 2023) evaluates different formats of prompts to find the best performance combination for the text-to-SQL task.

However, existing LLM-based methods entirely rely on human-labeled demonstrations, and demand high labeling costs be adapted to a new domain. Therefore, we propose FUSED to synthesize text-to-SQL demonstrations based on LLMs using provided domain databases without human labeling, effectively reducing the labor cost.

5.2 In-Context Learning

In-context learning is an effective method to enhance the reasoning ability of LLMs by providing several demonstrations to guide reasoning (Xun et al., 2017). Some works propose to automatically select relevant demonstrations for each user question to improve the performance of LLMs (Zhang et al., 2023b; Shum et al., 2023). Another kind of work enhances in-context learning by synthesizing relevant fine-tuning data (Wang et al., 2023).

However, existing methods only demonstrate that increasing the diversity of the demonstrations can enhance performance but do not discuss if the diversity of the existing labeling data is sufficient, and how to increase the diversity of the demonstrations (Su et al., 2023; Levy et al., 2023). Therefore, we present a diversity measurement metric to show that the existing labeling data of the text-to-SQL task is insufficient and propose FUSED to enhance the diversity by iterative synthesis.

6 Conclusion

In this paper, we improve the performance of the text-to-SQL task using in-context learning from the perspectives of insufficient diversity and the high labeling overhead of the human-labeled demonstration pool. We first present a metric to measure the diversity of the demonstration pool, based on which we analyze the diversity insufficient of the existing human-labeled text-to-SQL data. Based on the above analysis, we present FUSED, which synthesizes demonstrations using LLMs, thereby lowering the human labeling overhead. Besides, our method synthesizes demonstrations in multiple iterations, where each iteration fuses the demonstrations of the previous iteration to obtain new demonstrations that are dissimilar from the generated demonstrations, effectively enhancing the diversity. We adapt our method to two mainstream text-to-SQL datasets: Spider and KaggleDBQA. Experiments show that FUSED brings an average improvement of 3.2% and 5.0% with and without labeling data, proving the effectiveness.

Limitations

FUSED has two limitations, including: *1*. About the encoding of the demonstration sample step, directly splice the user question and the SQL could not fully reflect the attributes of them. In future work, we will try to encode the question and SQL according to the attributes separately. *2*. For the synthesized demonstration pool, we only improve the diversity, while ignoring the effect of the scale on the demonstration selection. Our future work will filter the synthesized results, reducing the scale of synthesis under the premise of ensuring diversity.

Ethics Statement

All datasets and models used in this paper are publicly available, and our usage follows their licenses and terms.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ján Cegin, Branislav Pecher, Jakub Simko, Ivan Srba, Mária Bieliková, and Peter Brusilovsky. 2024. Effects of diversity incentives on sample diversity and downstream model performance in llm-based text augmentation. *ArXiv*, abs/2401.06643.

Shuaichen Chang and Eric Fosler-Lussier. 2023a. How to prompt LLMs for text-to-SQL: A study in zero-shot, single-domain, and cross-domain settings. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

Shuaichen Chang and Eric Fosler-Lussier. 2023b. Selective demonstrations for cross-domain text-to-SQL. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14174–14189, Singapore. Association for Computational Linguistics.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *ArXiv*, abs/2308.15363.

Tonglei Guo. 2023. The re-label method for data-centric machine learning. *ArXiv*, abs/2302.04391.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273, Online. Association for Computational Linguistics.

Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1401–1422, Toronto, Canada. Association for Computational Linguistics.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Chenhao Ma, Kevin C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *ArXiv*, abs/2305.03111.

Zhenwen Li and Tao Xie. 2024. Using llm to select the right sql query from candidates. *ArXiv*, abs/2401.02115.

Man Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. 2024. In-context learning with retrieved demonstrations for language models: A survey. *ArXiv*, abs/2401.11624.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing text-to-SQL capabilities of large language models: A study on prompt design strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14935–14956, Singapore. Association for Computational Linguistics.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, Fei Huang, and Yongbin Li. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *ArXiv*, abs/2208.13629.

Srikumar Ramalingam, Daniel Glasner, Kaushal Patel, Ravi Vemulapalli, Sadeep Jayasumana, and Sanjiv Kumar. 2021. Less is more: Selecting informative and diverse subsets with balancing constraints. *ArXiv*, abs/2104.12835.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, I. Evtimov, Joanna Bitton, Manish P Bhatt, Cristian Cantón Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre D'efossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950.

Kashun Shum, Shizhe Diao, and Tong Zhang. 2023. Automatic prompt augmentation and selection with chain-of-thought from labeled data. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12113–12139, Singapore. Association for Computational Linguistics.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Selective annotation makes language models better few-shot learners. In *The Eleventh International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Guangxu Xun, Xiaowei Jia, Vishrawas Gopalakrishnan, and Aidong Zhang. 2017. A survey on context learning. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):38–56.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023a. ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, Singapore. Association for Computational Linguistics.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023b. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

SQL Synthesize
Synthesize one SQL query for the given database.

{database}
– Synthesize a new single SQL for the above database imitating {SQL1} and {SQL2}.
SELECT

Table 6: The prompt for the SQL synthesis.

A How to Calculate Diversity Measurement

$$\begin{array}{ll} \max\limits_{u} & \min_{d_i \in D} \texttt{dist}(u, d_i) \\ \text{s.t.} & u \in \texttt{Convex}(D) \end{array} \tag{A.1}$$

As the discussion in §2, we define the diversity measurement as $(\max_u \min_{d_i \in D} \texttt{diff}(u, d_i))^{-1}$. The process of calculating the diversity measurement can be formulated as Equation A.1, where $\texttt{dist}(u, d_i)$ denotes the Euclidean distance between the embedding vectors corresponding to the user question and the demonstration, and $\texttt{Convex}(D)$ denotes the convex hull of the demonstrations.

We use `dist` to represent `diff` since the closer the distance between the embedding question and the embedding demonstration, the more similarity between the question and the demonstration. The user question u should be in the area surrounded by the convex corresponds to the question-related domain, and the user questions are highly related to the domain and have a high probability of locating in the convex.

We use SciPy (Virtanen et al., 2020) to solve Equation A.1. Since the solution of this optimization process is greatly affected by the initial value, we repeatedly sample the initial value until the difference between the result and the previous maximum value is less than $1e{-}3$.

B Synthesize Text-to-SQL Demonstrations with LLMs

In this section, we discuss how to employ LLMs to obtain the initial demonstration pool with the given database, lowering the human-labeled overhead. The prompts we used are shown in Appendix C.

SQL Synthesize Following the previous work (Chang and Fosler-Lussier, 2023b), we synthesize SQL based on the linearized schema of the given database with LLMs. During synthesis, we ask LLMs to generate multiple SQLs for each database to enhance the diversity of the results with the sampling generation. The prompt we used is shown in Table 6.

Question Synthesize
Using natural language, generate a question corresponding to the given SQL.
Different examples are separated with '\n\n'.

{demonstration1}

{demonstration2}

{database}
– Using natural language, generate a question corresponding to the given SQL: {SQL}.
Question:

Table 7: The prompt for the question synthesis.

Question Synthesize We synthesize the corresponding questions of the generated SQL with the linearized schema of the. We first synthesize SQL instead of questions because LLMs could generate questions that are hard to answer using SQL (Cheng et al., 2023), and it is harder to validate the semantic consistency between the SQL and the question for generating questions first. The prompt of this step is shown in Table 7.

Validate Due to the limitation of the model performance, it is hard to guarantee that the semantics of all synthesized SQL-question pairs are completely consistent, resulting in a decrease in the quality of the synthesized demonstration. To improve the quality of the synthesized results, we verify the semantic consistency between the synthesized questions and SQL. We generate SQL based on the question and then evaluate if the generated SQL is the same as the synthesized SQL, for which we use LLMs to reduce the cost of fine-tuning. The prompts for text-to-SQL follow Chang and Fosler-Lussier (2023b).

C Prompts

The prompts of the SQL generation and the question generation are shown in Table 6 and Table 7. The formats of `{database}` and `{demonstration}` are same as Chang and Fosler-Lussier (2023b).

D Baseline Model

CodeLlama CodeLlama is a model based on Llama2 (Touvron et al., 2023), which is fine-tuned on a large amount of code data and can better solve code-related problems (including SQL).

GPT3.5 GPT3.5 is an improved model based on GPT3 (Brown et al., 2020), which further enhances performance through additional task-specific fine-tuning. We use Azure OpenAI API of `gpt-3.5-turbo` of GPT3.5 for our experiments [3].

ACT-SQL ACT-SQL (Zhang et al., 2023a) is a method to construct the chain-of-thought rationales based on SQL automatically. This method synthesizes reasoning steps with table names, column names, and values used in the SQL.

ODIS ODIS (Chang and Fosler-Lussier, 2023b) is an automatic demonstration selection method designed for the text-to-SQL task. This method selects out-domain demonstrations from the labeled data and synthesizes in-domain demonstrations based on the databases related to the user question.

E Settings of Analysis Experiments

We adapt analysis experiments under the setting of:

CodeLlama-34b CodeLlama is one of the most mainstream code generation models at present, which achieves near the performance of the closed-source model (as shown in Table 1) in the open-source model with less inference cost (no need to call API), of which CodeLlama-34b is the best performance in this series of models.

Evaluating without values Regarding the text-to-SQL task, current research mainly focuses on how to generate SQL with the correct structure, while paying less attention to extracting the condition values exactly, since this requires the memorizing ability rather than the semantic parsing ability.

[3] https://azure.microsoft.com/en-us/products/cognitive-services/openai-service